Parte 1

1. ¿Qué es Java y cuál es su importancia en el desarrollo de software?

Es un lenguaje de programación orientado a objetos multiplaforma, es decir, puede ejectuarse en cualquier sistema operativo a través de un máquina virtual java. Cuenta con una de las comunidades que facilitan el desarrollo de nuevas librerias y frameworks , actualmente mantiene una liberación de nuevos releases cada 6 meses, lo que permite que el lenguaje pueda seguir evolucionando. Es un lenguaje ampliamente soportado por los provedores de Nube , y por su versatilidad, es utilizados en la construcción de Apis para multiples propositos y tecnologias.

2. ¿Qué es una clase en Java y cuál es su relación con los objetos?

Una clase es un boceto que permite la representación de una entidad ; abstrae las caracteristicas de una entidad del mundo real mediante la declaración la clase y sus los elementos como pueden ser atributos , métodos , constructores. El objeto es la creacíon de una clase, usualmente , a esta acción se conoce como instanciar una clase, proceso mediante el cual se crea y se asigna una dirección en la memoria mediante el uso del constructor de la clase o mediante la inyección de dependencias a través del uso de algún framework. Ejm Car car = new Car(“2013”, “owner”)

3. ¿Qué es un método en Java y cuál es su propósito?

Un metodo encapsula un conjunto de instrucciones con un próposito en particular , esta compuesto por una firma , la cual especifica los parametros de entrada y de salida, el nombre del método y la implementación de la lógica que va ejecutar el método. El uso de método tiene diversos propositos cómo la reutilización , facilidad de lectura, separación de dependencias.

4. Explica qué es la herencia en Java y proporciona un ejemplo.

Herencia en java es un proceso mediante el cual una clase, denominada subclase puede adquirir caracteristicas y comportamiento de una la clase padre.

class Car {

private String anio;

}

class Toyota extends Car {

public Toyota() {

super();

}

}

5. ¿Qué es una interfaz en Java y cómo se diferencia de una clase abstracta?

Una intefaz y una clase abstracta se utilizan para obtener abstracción al momento de declarar firmas. Una clase abstracta contiene el keyword abstract en su declaración, mientras que la interfaz es un boceto que suele ser implementado en una clase . La interfaces puede contener metodos abstractos como no abstractos, mientras que la interfaz solo metodos abstractos . Las interfaces tambien puede contener un metodo con implementación pero deben ser declarados de manera estatica, sus variables deben ser finales y estaticos. Mientras que en una clase abstracta pueden ser estaticos, no estaticos, no finales.

6. Describe qué es Spring Boot y cuál es su objetivo principal en el desarrollo de aplicaciones.

Spring boot es un maquina de generación de beans que permite gestionar de manera eficiente los módulos de Spring de versiones anteriores o versiones nuevas. Poder conectar o desconectar los diversos modulos de spring framework es la caracteristica que lo hace verstail en comparación del framework tradicional. El objetivo priincipal es que su versatilidad y facilidad de uso permite reducir el tiempo en construcción y configuración de una aplicación.

La inversión de control e inyección de dependecias se realiza de manera mas sencilla gracias a los starters, por ejemlo el starter de jpa permite la configuración y creación de objetos para la conexión base de datos como datasource de una manera mas simplificada. El agrupamiento idoneo de los starter facilita la construcción de un Api, no preocupaciones por la compatibilidad entrer versionas de una librerias u otra, no es necesario validar la compatibilidad mientras se use el starter parent correcto, ofrece herramientas para el autoreloading de una aplicación , servidores embebidos para su facil despliegue, contiene starters para la verificación del estado del servicio

7. ¿Cuál es la diferencia entre una aplicación Spring Boot y una aplicación Java tradicional?

El framework Spring boot permite la construcción de Apis o librerías a través de la reutilización de librerías o uso de las características del framework, en cierto grado, la tarea de configuración y creación de objetos es delegada al framework mientras que el desarrollo exclusivamente con Java implicaría que la gestión , configuración y construcción sea una tarea realizada desde Cero.

8. ¿Qué es la inyección de dependencias en Spring Boot y por qué es importante?

La inyección de dependecias es un concepto que se ha creado en las tecnologias de Spring boot, en este paradigma es el framework quien se encarga y controla la creación de los objetos java, ya sea desde el propio codigo fuente utilizando las anotaciones , injeccion por constructor o a través de la inversion de control , mediante el cual se instancian objetos a partir de la declaración en un archivo de propuedad o Yaml

9. ¿Qué es un Bean en Spring y cómo se configura en una aplicación Spring Boot?

Spring boot es un creador y orquestador de beans. En el contexto de Spring , un bean es toda instancia de un objeto Java creado por el framework. La creación o configuración de un bean ocurre atraves de la inyección de dependencias por constructor o por argumentos, la utilización de anotaciones como autowired, o la creación de instancias a partir de archivos de propiedades o yaml , mediante la inversión de control

10. ¿Cuál es la función principal de Spring MVC y cómo se relaciona con Spring Boot?

Spring Boot es el orquestardor customizado de módulos y dependencias, por tanto puede utilizar los módulo de Spring MVC,

Spring MVC está orientado al desarrollo de aplicaciones web , por su versatilidad se puede utilizar para producir Api que sirven contenido web bajo el paradigma de un framework modelo, vista y controlador, también puede servir como Api que produce datos ya sea en el formato xml o json.

Parte 2

¿Qué es Docker y cuál es su propósito en el desarrollo de aplicaciones?

Docker es una tecnologia de containerización y virtualización que permite aislar y crear un recurso independiente y poder gestionar los recursos de un computador o servidor de manera independiente a través del demonio o servicio de docker . En el desarrollo de aplicaciones, en el contexto de aplicaciones Java permite poder construir , configurar y desplegar a través de directivas, variables y comandos que se especifican en un archivo Dockerfile

¿Cuál es la diferencia entre una imagen y un contenedor en Docker?

La imagen es un plantilla de carácter inmutable que define como se va materializar o crear un contenedor . Un contenedor es la instancia en ejecución , el cual es generado a partir de una Imagen Docker

Explica qué es un Dockerfile y cuál es su función en la creación de imágenes.

Un archivo Dockerfile hace posible poder configurar, construir y ejecutar un componente de software.

¿Qué es un registro de Docker (Docker registry) y cuál es su importancia en el ecosistema de Docker?

Docker registry es un sistema de versionamiento , almacenamiento y distribución de images de Docker . Utilizar un docker registry permite tener un control sobre las imágenes, en el contexto de aplicaciones cloud nativas, permite poder un desarrollo y delivery agil de artefactos, actuando como un repositorio compartidos de imágenes ,

¿Qué es Kubernetes y cuál es su objetivo principal en la administración de contenedores?

Kubernetes es una tecnologia que permite la orquestación de contenedores , se ha convertido en un estandar en la administración de contenedores , permite poder gestionar y desplegar aplicaciones , configurar el escalamiento de acuerdo a las necesidades, monitorear la ejecución de las aplicaciones de manera automatizada.

¿Cuál es la diferencia entre un pod y un contenedor en Kubernetes?

Un pod es una componente de ejecución en Kubernetes que suele agrupar 1 o mas contenedores

Describe qué es un servicio en Kubernetes y cómo se utiliza para exponer aplicaciones.

En Kubernetes un servicio es una capa de abstracción que permite definir un conjunto de pods, habilitar la exposición de trafico externo, balanseaminto de carga y service discovery para los pods pertenecientes a la red en cuestion

¿Qué es un nodo (node) en el contexto de Kubernetes y cuál es su función?

Es la representacion de una colección de recursos fisicos o virtuales, usualmente un Nodo contiene los servicios necesario para ejecutar pods, comunicación con el control plane, configuración de red y la ejecución de asignamiento de cargas

Explica qué es un manifiesto de Kubernetes (Kubernetes manifest) y cómo se utiliza para definir la configuración de una aplicación.

El manifiesto en Kubernetes es un archivo que define la configuración de diversos componentes. De acuerdo a la especificación puede ser escrita en el formato Json o Yaml , nos permite especificar el estado deseado de un objeto en Kubernetes , algunos ejemplos son PodSpec, Deployment, Network policy. Usualmente se suele utilizar el comando kubectl apply -f transaction-pod.yaml

¿Cuál es la importancia de los volúmenes persistentes en Kubernetes y para qué se utilizan?

En el contexto de Kubernetes, la de data de una aplicación ejecutandonse en un pod al momento de eliminarse o recrearse el pod desaparece. La importancia de los volumenes persistenes es que permiten que los datos puedan persistir independientemente del ciclo de vida del pod . Por ejmplo al momento de crear un aplicación Java, que escribe un archivo pdf , durante la ejecución podria eliminar el pod, sin embargo, con la definición de un volumeMount se puede especificar una ruta donde puedan salvarse los cambios del archivo.